Meaning-Text Unification Grammar: modularity and polarization

Sylvain Kahane,
Modyco, U. Paris 10 /
Lattice, U. Paris 7
sk@ccr.jussieu.fr

François Lareau
OLST, U. de Montréal /
Lattice, U. Paris 7
francois.lareau@umontreal.ca

This article presents the Meaning-Text Unification Grammar's current state, now that its formal foundations have been clarified with the development of Polarized Unification Grammar. Emphasis is put on the model's architecture and the role of polarization in linking its modules — semantics-syntax interface and the well-formedness grammars of each representation level.

1. Introduction

This article presents the current state of Meaning-Text Unification Grammar [MTUG], building on the recent development of Polarized Unification Grammar [PUG]. PUG can simulate most formalisms that are based on structure unification, such as rewriting grammars, TAG, LFG or HPSG (Kahane, 2004). However, although PUG was initially developed as a formal basis for MTUG, how it can actually be used for writing an MTUG has never been discussed.

MTUG's architecture is based on the Meaning-Text Theory [MTT] (Mel'cuk 1988; Kahane 2001b). Fewer levels of representation are considered in the current version of MTUG: semantic, syntactic, morphotopological and phonological. Each one has its own grammar. A semantic structure is a graph, a syntactic structure is a dependency tree, a morphotopological structure is an ordered tree (Gerdes & Kahane 2001) and a phonological structure is a chain. On these can be added other structures such as a logical or communicative structure at the semantic level or else a prosodic structure at the phonological level, but we will not discuss these points here. These representation levels are ordered (semantic < syntactic < morphotopological < phonological), which yields three interface modules: semantics-syntax, syntax-morphotopology and morphotopology-phonology).

MTUG therefore requires a formalism that can handle several types of structures (graph, dependency tree, ordered tree, chain) and link them. As we will see, PUG is a mathematical formalism tailored specifically for handling such structures as simply as possible. It relies on polarities that have to be neutralized for driving sentence synthesis or analysis in a way similar to how atoms' valence controls molecule formation. Our implementation of MTUG as a PUG makes extensive use of polarization: each module (well-formedness grammars as well as interface grammars) controls the construction of objects and the saturation of structures only via polarization. Each has its own polarity and uses the polarity of the adjacent modules to control its interaction with them. The order in which those polarities are neutralized induces a particular procedure. Neutralizing all polarities of a given level before considering those of the next level induces a sequential synthesis/analysis procedure (the one typically used in classic MTT implementations), whereas neutralizing all the polarities introduced for a given word before considering other polarities allows for an incremental synthesis/analysis procedure, despite a stratified architecture.

Due to the lack of space, we will leave aside the morphotopological and phonological levels as well as their interfaces, and we will focus only on the semantic and syntactic structures and the semantics-syntax interface (see Kahane & Lareau 2005, for a brief discussion of the morphotopological structure and its interface with syntax in PUG). We will briefly introduce PUG in section 2. Section 3 will introduce well-formedness grammars for the structures considered in this paper. Finally, in section 4 we will show how the PUG formalism can be used for writing MTUG interface grammars.

2. Polarized Unification Grammars [PUG]

Polarized Unification Grammars are devices that generate sets of finite structures. A structure is made of *objects*. For instance, an oriented graph is made of *nodes* and *arrows*. Each arrow is linked to two nodes by the functions *source* and *target*. It is these *functions* that provide the structure proper.

A *polarized structure* is a structure whose objects are polarized, *i.e.* linked through a function to a value that belongs to the finite set P of *polarities*. The set P is provided with a commutative and associative operation called *product* (noted " . "). A subset N of P contains *neutral polarities*. A polarized structure is said *neutral* if all of its objects are neutral. We will use in this paper a set P = {●,○,●} (we will call these polarities:

● = black = saturated, ○ = white = obligatory context and ● = grey = absolute neutral), with N = {●,●}, and a product defined in the following table (⊥ indicates that two polarities cannot combine):

·	●	○	●
●	●	○	●
○	○	○	●
●	●	●	⊥

Table 1. The product of polarities

Structures are combined by *unification*. Unifying two structures A and B gives a new structure A⊕B by matching some of the objects of A with some of the objects of B. When A and B are unified, the polarity of an object of A⊕B that was obtained by matching an object of A with one of B is the product of the polarities of these two objects. All functions of objects that are unified must themselves be unified (just as the features when we unify feature structures).

A *polarized unification grammar* [PUG] is defined by a finite set T of object types (with functions attached to each kind of object type), a polarity system (P,.), a neutral polarities subset N of P and a finite set of elementary polarized structures, whose objects are described by T and of which one can be marked as an initial structure. The structures *generated* by the grammar are the neutral structures obtained by combining the initial structure and/or a finite number of elementary structures. This formalism is monotonic (with the following order on polarities: ● < ○ < ●) and structures can be combined in absolutely any order (due to the commutativity and associativity of (P,.)).

We will introduce in the next section two examples of PUG that respectively generate semantic graphs and syntactic trees. Examples throughout this paper will be based on (1):

(1) *Peter ate two apples.*

3. Well-formedness grammars

3.1. Semantic grammar

Our semantic representations are based on a graph of predicate-argument relations between semantemes (*i.e.* meanings of lexical and

grammatical units of a sentence). One could superimpose on it other kinds of structures that encode various types of information, such as an information structure or a logical scope structure (Polguère 1992, Mel'cuk 2001, Kahane 2005). The nodes of a semantic representation encode semantemes and the edges represent predicate-argument relations. Figure 1 below shows a simplified fragment of $\mathcal{G}_{\text{sem}}$ — a graph grammar which generates the semantic representation of (1). Each of its objects has a polarity, noted p_{sem} in the following discussion[1], that indicates which objects are actually constructed by the rule (objects polarized in black) and which only give the semantic valence of predicates (objects polarized in white).

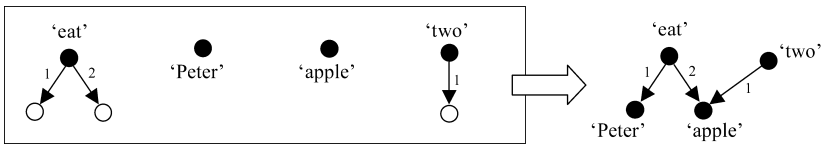

Fig. 1. A fragment of $\mathcal{G}_{\text{sem}}$ generating the semantic representation of (1)

3.2. Syntactic grammar

Our syntactic representations are dependency trees whose nodes are labeled with the name of a lexeme and are linked to other objects representing the grammemes via so-called *grammatical* functions [2]. The edges are labeled with syntactic functions (which are language-specific). The rightmost part of Figure 2 shows the syntactic representation of (1).

Our syntactic grammar $\mathcal{G}_{\text{synt}}$ uses a double polarization, each verifying a different property of syntactic representations. The polarity p_{synt} constructs all the objects and checks that each of them has all the functions required by its type: *label*, *source* and *target* for edges, part of speech (*pos*) for the nodes, and eventual grammatical functions for some

[1] We will refer to polarities by the name of the functions that associate them to the objects. Thus, any object of a rule in $\mathcal{G}_{\text{sem}}$ is associated to a function p_{sem} which returns as its value one of the polarities in the set P described above.

[2] In our figures, we indicate these functions in italic, followed by the grammemes. Grammemes are in turn linked to the lexemes through opposite functions. This double linking makes lexemes and grammemes inseparable since the unification of one forces the unification of the other. However, we will not discuss this mechanism here.

nodes (*number* for nouns, *mood*, *tense*, *number* and *person* for verbs, etc.) [3]. The polarity $p_{\text{synt-gov}}$, which we hide in our figures, is used to verify that the generated structure is a tree (*i.e.* that every node except one has a unique governor).

Figure 2 shows a fragment of $\mathcal{G}_{\text{synt}}$ that generates the syntactic representation of (1). The very first rule in this figure is the initial structure, which has to be used once (and only once) and which corresponds to the root of the tree (it has a black $p_{\text{synt-gov}}$ polarity, hidden in Figure 2, indicating that it cannot have a governor). The following four rules are lexical rules; they give the part of speech of lexemes as well as the grammemes that are necessary to them. Note that these rules do not control the syntactic valence of lexemes, as it will be controlled by the semantics-syntax interface. The next three rules are sagittal rules; they describe various possible syntactic relations (they also control the tree structure by having a white $p_{\text{synt-gov}}$ polarity on the governor and a black one on the dependent — again, hidden in Figure 2). The other rules are grammatical rules. They may be context-sensitive: thus, the indicative requires a subject (and vice versa), a numeral imposes the plural on its governing noun, or a non pronominal subject implies the 3rd person on the verb. Note that the introduction of a grammeme might depend on another grammeme: the indicative, for instance, will impose a tense, while infinitive will not.

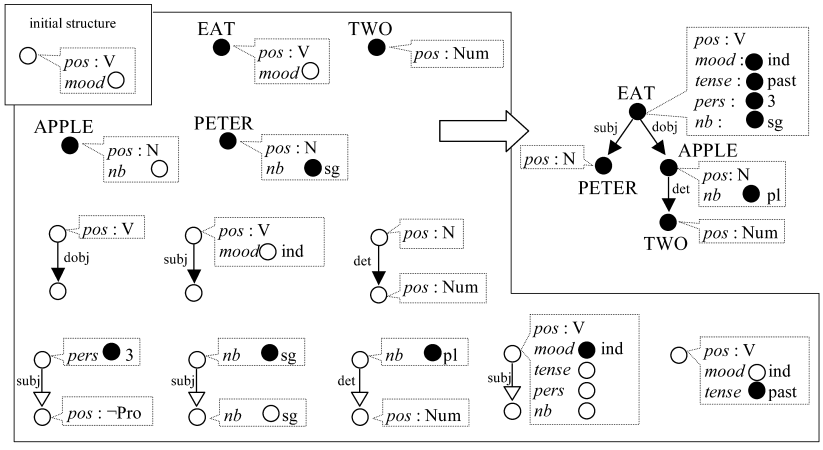

Fig. 2. A fragment of $\mathcal{G}_{\text{synt}}$ generating the syntactic representation of (1)

[3] We use a simplified grammatical system, leaving aside, for instance, definiteness for nouns or finiteness and voice for verbs.

4. Interfaces

4.1. Correspondence grammars

A correspondence grammar $\mathcal{G}$ is a grammar that establishes a correspondence between structures belonging to two sets, that we will call $\mathcal{A}$ and $\mathcal{B}$. The rules of $\mathcal{G}$ establish a correspondence between elementary structures that make up the elements of $\mathcal{A}$ and $\mathcal{B}$. There are three possible ways of considering $\mathcal{G}$: as a transductive, equative, or generative grammar, depending on whether it is given one, two or no structures as input (Kahane 2001a). A transductive grammar translates a given structure into another, an equative grammar verifies that two given structures correspond to each other, while a generative grammar creates a pair of structures that correspond to each other.

Let us consider the equative functioning of $\mathcal{G}$, which is the simplest: $\mathcal{G}$ partitions both structures in an equal number of fragments that are in a one-to-one correspondence. In terms of polarities, it means that both structures given as input have to be declared as resources that $\mathcal{G}$ must "consume" (thus ensuring that both structures have totally been put in a correspondence). The objects of both structures bear a white $p_{\mathcal{G}}$ polarity that the rules of $\mathcal{G}$ must neutralize. The rules of $\mathcal{G}$ contain corresponding objects (from the levels of representation of $\mathcal{A}$ and $\mathcal{B}$) that bear a black $p_{\mathcal{G}}$ polarity. Hence, by neutralizing the two given structures, $\mathcal{G}$ establishes a correspondence between them.

Let us now turn to the transductive functioning of $\mathcal{G}$. It is this mode that actually models linguistic synthesis or analysis. We give as input to $\mathcal{G}$ a structure A belonging to $\mathcal{A}$, all objects of which bear a white $p_{\mathcal{G}}$ polarity. This triggers a number of rules from $\mathcal{G}$ (which all contain at least one black element) in order to neutralize A. These rules also build a new structure B, synchronized with A. A well-formedness grammar of the structures of $\mathcal{B}$ must then verify that B is an actual element of $\mathcal{B}$. The structure B, while being neutral for $\mathcal{G}$ (all its objects now bearing a black $p_{\mathcal{G}}$ polarity), must force the application of the grammar of $\mathcal{B}$. In order to do so, all of its objects must also have a white $p_{\mathcal{B}}$ polarity. In sum, every module has its own polarity controlling the construction of the structures it describes, but in order to call the adjacent modules, the objects of $\mathcal{A}$ and $\mathcal{B}$ must have, besides their respective $p_{\mathcal{A}}$ and $p_{\mathcal{B}}$ polarities, a white $p_{\mathcal{G}}$ polarity. We also have to add to the rules of $\mathcal{G}$ a white $p_{\mathcal{A}}$ polarity for the objects of the level of $\mathcal{A}$ and a white $p_{\mathcal{B}}$ polarity for the objects of the level of $\mathcal{B}$. Each object constructed by $\mathcal{A}$ will therefore have a double polarization $p_{\mathcal{A}}$-$p_{\mathcal{G}}$ (●,○), while the corresponding elements constructed

by $\mathcal{G}$ will have a double polarization $p_{\mathcal{A}}$-$p_{\mathcal{G}}$ (○,●). This yields a system with four polarities {(○,○),(○,●),(○,●),(●,●)}, equivalent to the system {○,—,+,●} of Bonfante *et al.* 2004 and Kahane 2004. These couples of polarities are called *articulation polarities*.

4.2. Semantics-syntax interface

The semantics-syntax interface $\mathcal{I}_{\text{sem-synt}}$ is a correspondence grammar between the set of semantic representations described by $\mathcal{G}_{\text{sem}}$ and the set of syntactic representations described by $\mathcal{G}_{\text{synt}}$. While discussing this interface we will illustrate its transductive mode in synthesis. It then takes as input a semantic representation similar to the resulting structure in Figure 1, except for that each object will bear a double polarity p_{sem}-$p_{\text{sem-synt}}$ (●,○) or (○,○). The objects of $\mathcal{I}_{\text{sem-synt}}$ all bear a $p_{\text{sem-synt}}$ polarity (white or black, depending on whether or not they are constructed by $\mathcal{I}_{\text{sem-synt}}$), but those of the semantic level (resp. syntactic) will also have a white p_{sem} polarity (resp. p_{synt}). Figure 3 shows a fragment of the semantics-syntax interface grammar. Only the $p_{\text{sem-synt}}$ polarity is shown here (all p_{sem} and p_{synt} being white). The doted lines are objects representing the correspondence between objects of adjacent levels and their polarity is shown by the rhombs that are superimposed on them.

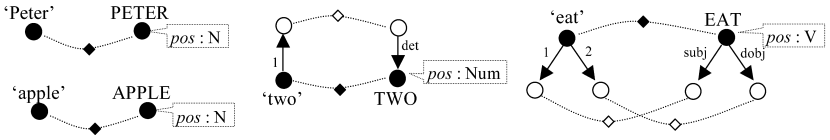

Fig. 3. A fragment of the interface grammar $\mathcal{I}_{\text{sem-synt}}$

In Figure 4, we give three structures: the semantic representation of (1) with the double p_{sem}-$p_{\text{sem-synt}}$ polarization, the result after $\mathcal{I}_{\text{sem-synt}}$ has neutralized all the polarities that it could (*i.e.* $p_{\text{sem-synt}}$ and p_{sem}, but not p_{synt}), and finally the result after applying $\mathcal{G}_{\text{synt}}$ on the syntactic tree constructed by $\mathcal{I}_{\text{sem-synt}}$. The polarities that allow the interaction with the adjacent module are shown as backgrounded. Note that agreement grammemes (such as *person* for verbs) do not have a direct semantic counter part, therefore they must not have a $p_{\text{sem-synt}}$ polarity.

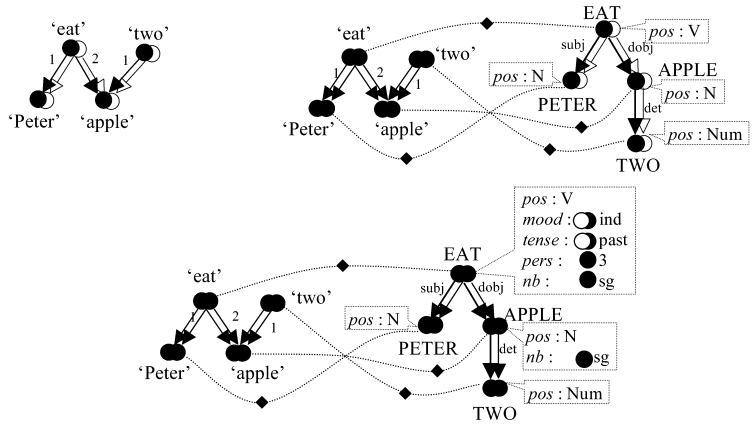

Fig. 4. The semantic representation of (1)
before and after the application of $\mathcal{I}_{\text{sem-synt}}$ and $\mathcal{G}_{\text{synt}}$

Just as $\mathcal{G}_{\text{sem}}$, the syntactic grammar $\mathcal{G}_{\text{synt}}$ must be enriched in order to interact with both the semantics-syntax interface and the syntax-morpho-topology interface. As the last structure of Figure 4 shows, applying $\mathcal{G}_{\text{synt}}$ on the output of $\mathcal{I}_{\text{sem-synt}}$ neutralizes all the p_{synt} polarities of the syntactic objects created by $\mathcal{I}_{\text{sem-synt}}$. However, it is not possible to introduce the lexeme EAT without assigning a mood to it (*cf.* Figure 2), and if we assign the indicative, for instance, we are then forced to use a tense, as well as agreement grammemes for person and number. The grammar $\mathcal{G}_{\text{synt}}$, while neutralizing the p_{synt} polarities of the output of $\mathcal{I}_{\text{sem-synt}}$, introduces grammemes that were not in this output and which (unless they are agreement grammemes) bear a white $p_{\text{sem-synt}}$ polarity (see Figure 4). These mood and tense grammemes must then be neutralized by rules of $\mathcal{I}_{\text{sem-synt}}$. However, there is no semanteme in the semantic representation given as input that could correspond to these grammemes. This is not a problem with MTUG, since the formalism makes it possible to modify on the fly the input representation in order to satisfy constraints that come from other levels of representation. Let us take for granted that our model includes the two rules shown in Figure 5. The first one is a rule of $\mathcal{G}_{\text{sem}}$ that gives a simple representation of one of the meanings of the past tense. The second rule belongs to $\mathcal{I}_{\text{sem-synt}}$ and establishes the correspondence between this meaning and the past tense grammeme. These two rules can be applied in the opposite direction from the undergoing synthesis process and modify the initial semantic structure, adding to it the meaning of the past

tense (we leave aside the mood for the sake of simplicity). We then obtain the last structure in Figure 5.

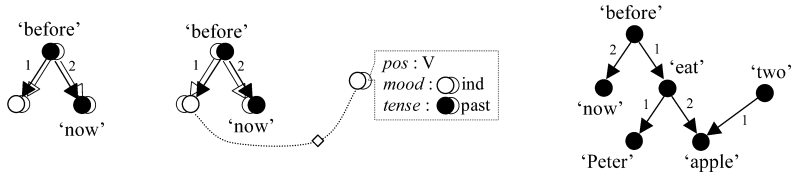

Fig. 5. The past tense

Of course, there are other rules that could satisfy the needs for a mood and a tense, and one would obtain as many modified semantic structures as there are ways to satisfy these syntactic constraints. A text generation system would then have to choose, among all the semantic structures proposed, which is the most suitable according to what it has in its world-knowledge database. In the case discussed here, such a system would have to answer the question "When does Peter's eating two apples take place: in the present, in the past or in the future?". This mechanism modelizes in an elegant way the fact that the expression of grammatical meanings is not necessarily triggered by a communicative need from the speaker, but might very well be imposed by the language (see Polguère 2000). This kind of mechanism, in addition to being plausible from a cognitive point of view, might prove particularly useful for machine translation, where the semantic representation obtained by analysis from the source language is not always directly expressible in the target language. It might as well be useful for analysis. It is then the sound chain that is the input and which can be modified if it is not totally well-formed. This might be a useful workaround for problems related to noise or performance flaws from human speakers. However, these hypotheses have not yet been validated through experimentation.

Conclusion

This article mainly focuses on the architecture of a linguistic model. We suggest a modular model (two well-formedness grammars and one interface grammar in the simplified version presented here, but a full-fledged model would include four well-formedness grammars and three interface grammars) that can handle several levels of linguistic representation and several types of structures with a single formalism.

This makes it possible to combine the rules in absolutely any order and allows the various modules to constantly interact with one another. This interaction and the saturation of structures are driven by the (generally multiple) polarization of every object. Each polarity born by an object having a well defined role, two types of algorithms can be considered: neutralizing the polarities specific to one module before considering other polarities would force the modules to apply in a sequential order, while neutralizing all the polarities associated with one object would force each element of the input structure to be processed from semantics to phonology (vice versa in the context of analysis). This second type of algorithm is possible only because one single formalism is used for all the modules and interfaces. We have also shown that polarities allow our model to accept as input underspecified structures lacking inflectional meanings, and still generate correct sentences by adding the missing semantemes to the initial representation. Finally, as Bonfante *et al*. (2004) have shown, polarization allows an efficient filtering of rules by reducing the amount of dead-end computing.

A model of French is currently under development. The heart of French topology has been described by Gerdes & Kahane (2004), the syntax of French tenses is described by Lareau (2004), other aspects of the semantics-syntax interface (mismatches between syntax and semantics and scope of quantifiers) are discussed by Kahane (2001b, 2005). An implementation of MTUG in the PUG formalism is also under process. More than coverage, we put emphasis on theoretical elegance and adequacy of formalization.

References

Bresnan J. (1999), *Lexical-Functional Syntax*, Blackwell, 1999.

Bonfante G., Guillaume B. & Perrier G. (2004), Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation, *Proceedings of CoLing,* Genève*,* 2004. P. 303—309.

Gerdes K. & Kahane S. (2001), Word order in German: A formal dependency grammar using a topological hierarchy, *Proceedings of ACL*, Toulouse, 2001. P. 220—227.

Gerdes K. & Kahane S. (2004), L'amas verbal au cœur d'une modélisation topologique du français, *Proceedings of Journées de la syntaxe — Ordre des mots dans la phrase française, positions et topologie*, Bordeaux, 2004. 8 p.

Kahane S. (2001a), What is a natural language and how to describe it? Meaning-Text approaches in contrast with generative approaches, *Computational Linguistics, Proceedings of CICLing 2001*, Mexico, Springer Verlag, 2001. P. 1—17.

Kahane S. (2001b), A fully lexicalized grammar for French based on Meaning-Text theory, *Computational Linguistics, Proc. CICLing 2001*, Mexico, Springer Verlag, 2001. P. 18—31.

Kahane S. (2004), Grammaires d'unification polarisées, *Proceedings of TALN*, Fès, 2004. P. 233—242.

Kahane S. (2005), Structure des représentations logiques polarisation et sous-spécification, *Proceedings of TALN.* 2005. 10 p.

Kahane S. & Lareau F. (2005), Grammaire d'Unification Sens-Texte : modularité et polarisation, *Proceedings of TALN.* 2005. 10 p.

Lareau F. (2004), *Vers un modèle formel de la conjugaison française dans le cadre des grammaires d'unification Sens-Texte polarisées*, Pre-doctoral exam report, Montréal, Université de Montréal, 2004.

Mel'cuk I. (1988), *Dependency Syntax: Theory and Practice*, Albany, SUNY Press, 1988.

Mel'cuk I. (2001), *Communicative Organisation of Natural Language*, Benjamins, 2001.

Nasr A. (1995), A formalism and a parser for lexicalised dependency grammars, *4th Int. Workshop on Parsing Tecnologies*, State University of New York Press, 1995.

Polguère A. (1992), Remarques sur les réseaux sémantiques Sens-Texte, in A. Clas (éd.), *Le mot, les mots, les bons mots*, Presses de l'Université de Montréal, 1992.

Polguère A. (2000), A "natural" lexicalization model for language generation, in *Proceedings of SNLP'2000*, Chiangmai, Thailand, 10—12 May 2000.

Shieber S. M. & Schabes Y. (1990), Synchronous tree-adjoining grammars, *Proceedings of the 13th Int. Conference on Computational Linguistics.* Vol. 3. 1990. P. 253—258, Helsinki, Finland.